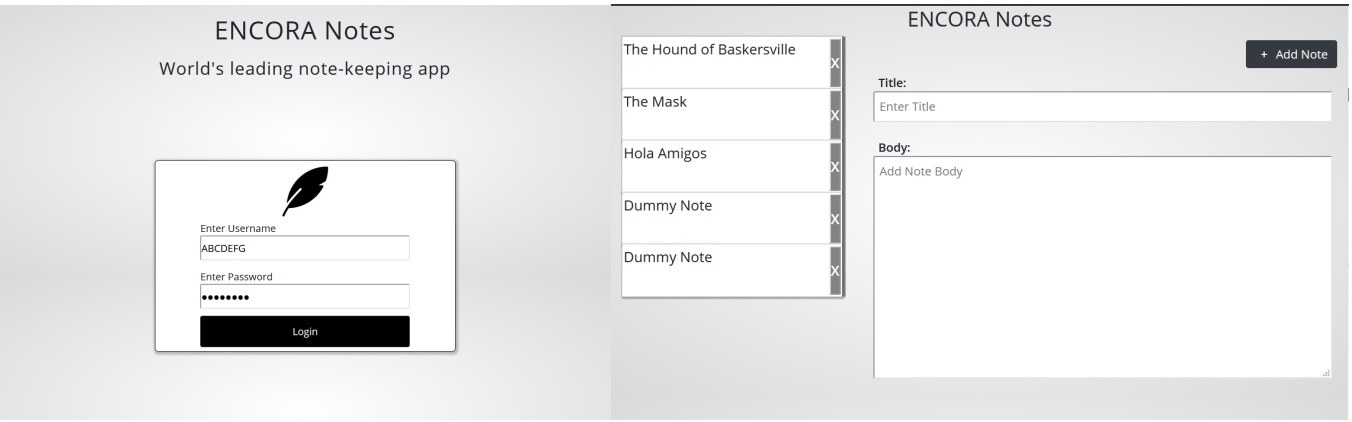
ENCORA Notes App Assignment README



Description: This is a note keeping app and has a complete front end and back end. LocalStorage is used for maintaining the session token of the user logged in.
Viewing, Retrieval of all notes, Addition, Deletion of notes was successfully achieved. Also, all these actions require authentication and cannot be done otherwise.

Author: Neel Shinde.

Front end tech used: **React + Redux (with Thunk), HTML, CSS, Bootstrap**
Backend tech used: **Node + Express, Cors, MYSQL, Body Parser**
All data seen on Front end is stored in the backend **MYSQL** DB.

NOTE: Scroll to the end for responsive mode screenshots - App is completely responsive.

Below are the details for each tech component and usage:

Database:
2 tables used:

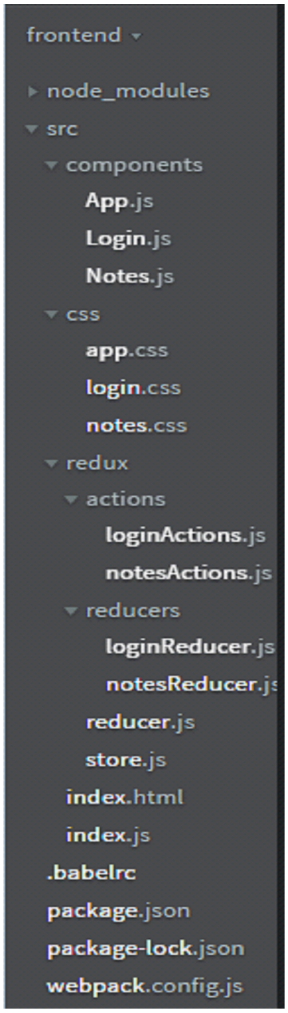
1) **users**: Has a list of usernames, password and authid (session ids)

username	password	authid
emma_green	eMMa_01	0
johndoe	john@123_	0
nicolekidman	sassy_01	0
steveb	steve00	steveb0gsziw5w4bvj
NULL	NULL	NULL

2) **notes** : Table of note titles, usernames, autoincrementing noteid and note body.

noteid	username	note	title
1	steveb	Once upon a time there was a Hound of Basker…	The Hound of Baskersville
2	steveb	Loren ISpum blah blah balh blah …….	The Mask
4	steveb	Hola Amigos! 2020 has been a bad year	Hola Amigos
NULL	NULL	NULL	NULL

React application is distributes in folder structure as below :

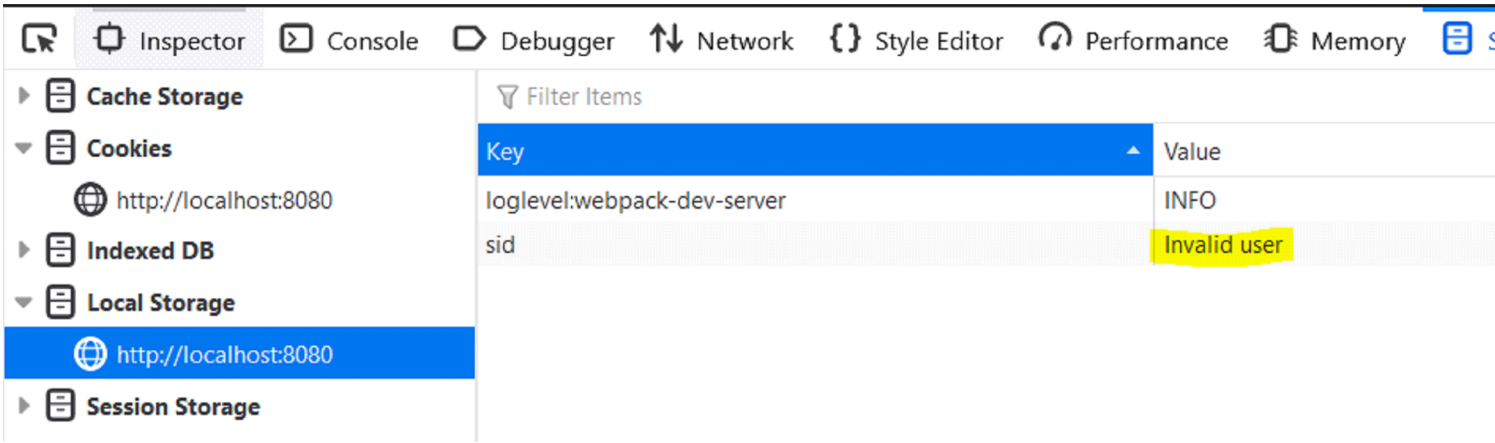


Backend simply has one file - **Server.js** that handles all requests.
The file has one comment before each request handler to know what request it is handling

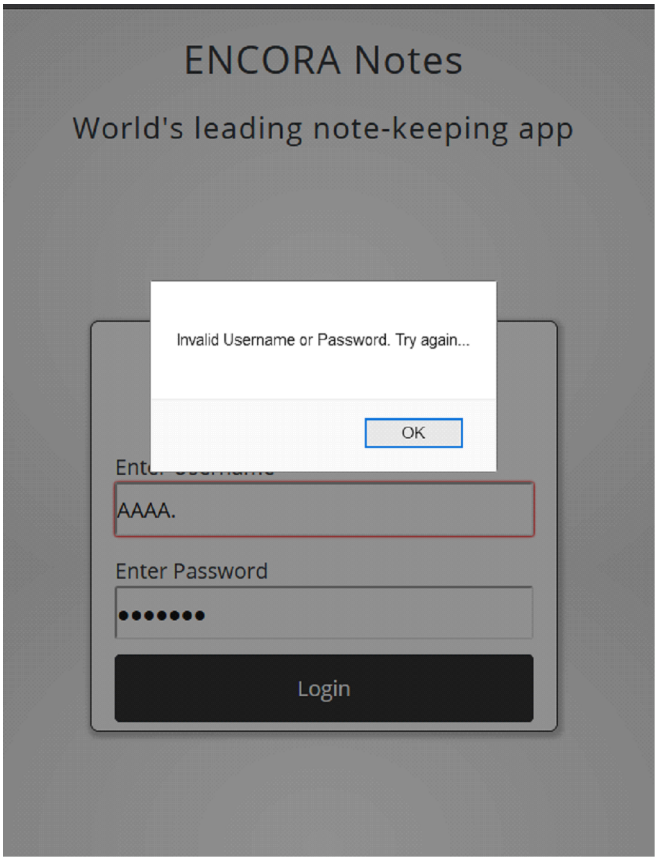
I have provided the necessary files on Github and in order to install the dependencies (node_modules), please run: npm install
Dev server command: npm run dev
Production build command: num run build
You can refer to the actual inner commands in the package.json file for clarity.

Working of the App:

Initial Session token:



Invalid attempt:

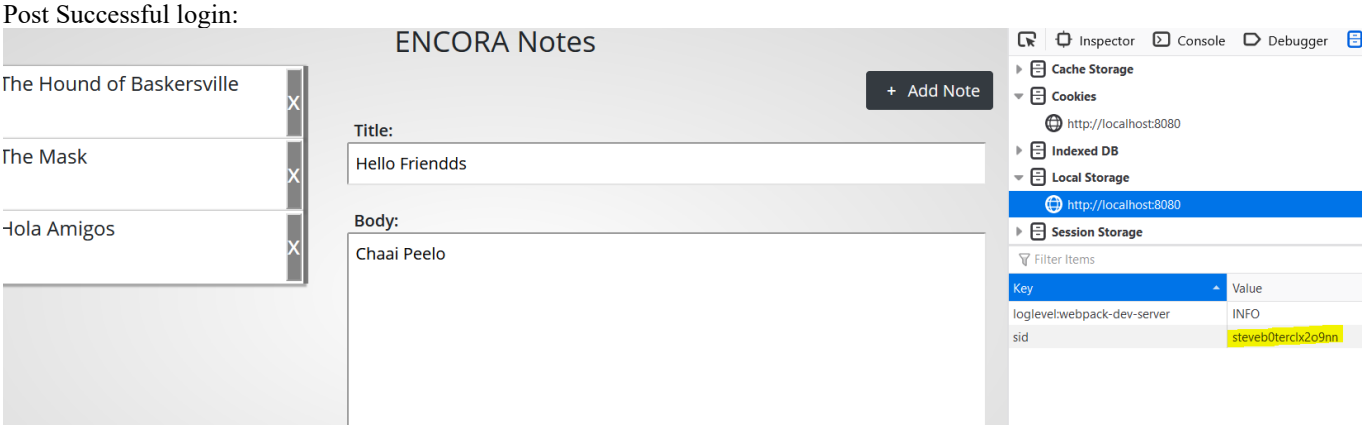


Successful login: Credentials used: user- steveb and pass- steve00

Username and Password is passed via the Authorization header as below: Authorization type - Basic

```
var xhttp = new XMLHttpRequest();
xhttp.open('POST', 'http://localhost:8000/auth', true);
xhttp.setRequestHeader('Content-Type', 'application/xml');
xhttp.setRequestHeader('Accept', 'application/notesapp');
xhttp.setRequestHeader('Authorization', 'Basic ' + btoa(props.username+':'+ props.password));
xhttp.send();
xhttp.onreadystatechange = () => {
    if(xhttp.readyState === XMLHttpRequest.DONE) {
        var status = xhttp.status;
        if (status === 0 || (status >= 200 && status < 400)) {
          if(xhttp.responseText.length > 7){
              console.log(xhttp.responseText)
              localStorage.setItem('sid', xhttp.responseText);
              window.location.reload(); // to refresh page so Notes page is loaded after checking session ID in local storage
          } else {
              console.log(xhttp.responseText);
              alert("No such user!");
          }
        } else {
          alert("Login attempt failed!");
        }
    }
}
```

Post Successful login:



DB Snapshot: session id matching the one from localstorage

```
1 • select * from users;
2
```

Result Grid | Filter Rows: | Edit:

username	password	authid
emma_green	eMMa_01	0
johndoe	john@123_	0
nicolekidman	sassy_01	0
steveb	steve00	steveb0terdx2o9nn
NULL	NULL	NULL

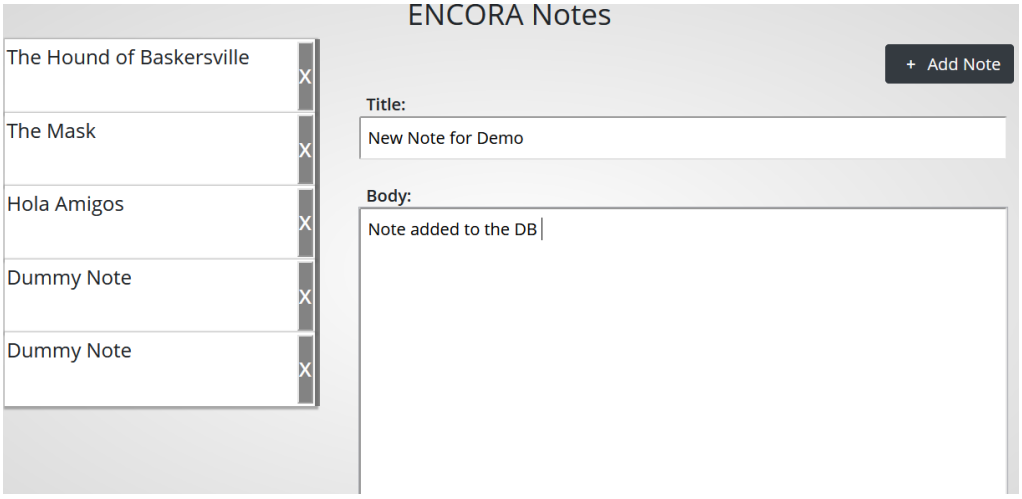
Server side nodemon log (from different successful login):

```
********** Login attempt ***************
select username from users where username = 'steveb' and password = 'steve00';
Sessionid : steveb0k48gilsy9b
************** Checking auth status *************
steveb0k48gilsy9b
select authid from users where authid = 'steveb0k48gilsy9b';
[ RowDataPacket { authid: 'steveb0k48gilsy9b' } ]
************** Getting note *************
steveb0k48gilsy9b
select authid from users where authid = 'steveb0k48gilsy9b';
select title, note from notes where username in (select username from users where authid = 'steveb0k48gilsy9b')
[
  RowDataPacket {
    title: 'The Hound of Baskersville',
    note: 'Once upon a time there was a Hound of Baskersville that turned out to be a case sollve by Sherlock Holmes.'
  },
  RowDataPacket {
    title: 'The Mask',
    note: 'Loren ISpum blah blah balh blah .......'
  },
  RowDataPacket {
    title: 'Hola Amigos',
    note: 'Hola Amigos! 2020 has been a bad year'
  }
]
```

Once logged in, the users will remain logged in irrespective of refreshs until the session token is present in the localstorage. Clear the session token to test another user.

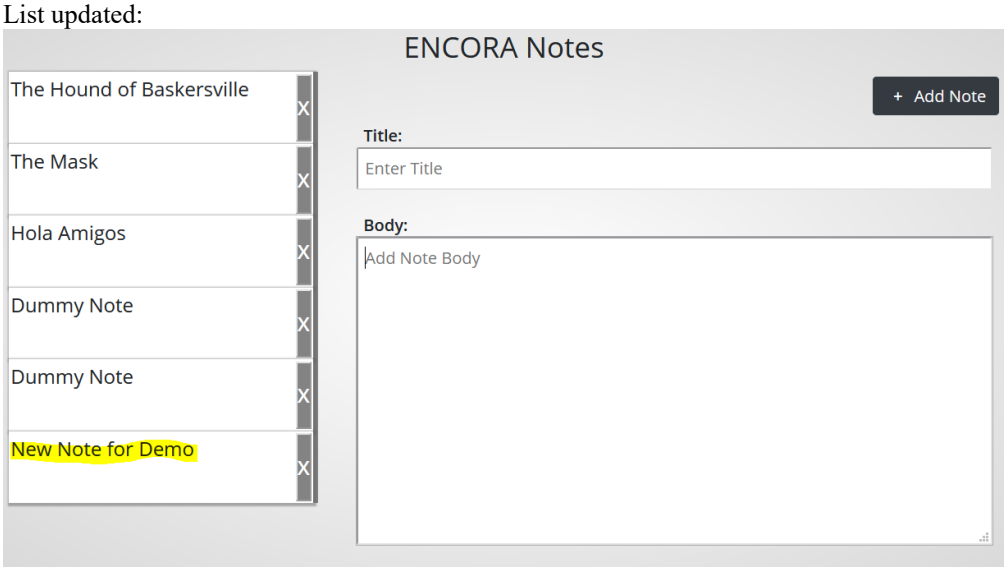
Addition of Note:
Before:

noteid	username	note	title
1	steveb	Once upon a time there was a Hound of Basker...	The Hound of Baskersville
2	steveb	Loren ISpum blah blah balh blah	The Mask
4	steveb	Hola Amigos! 2020 has been a bad year	Hola Amigos
20	steveb	Dummy Note for testing	Dummy Note
21	steveb	Dummy Note for testing	Dummy Note
NULL	NULL	NULL	NULL

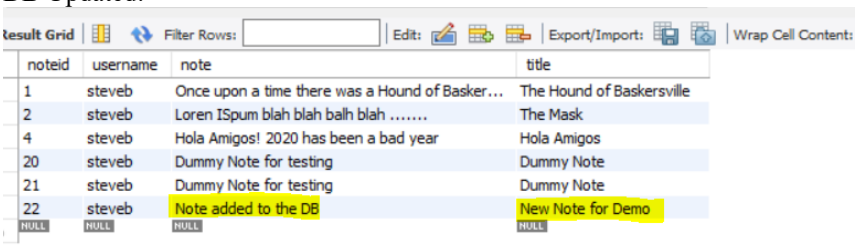


After:

List updated:



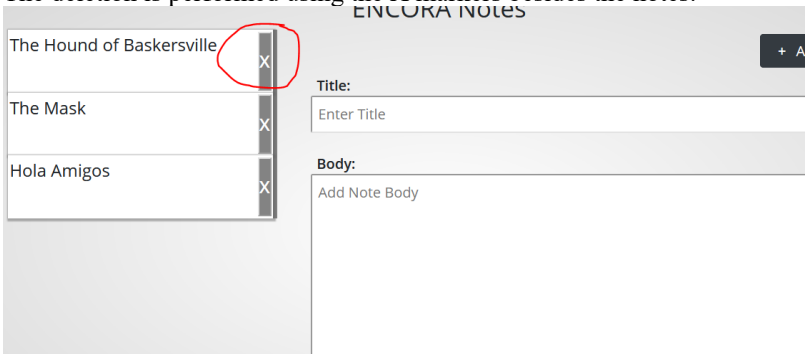
DB Updated:



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

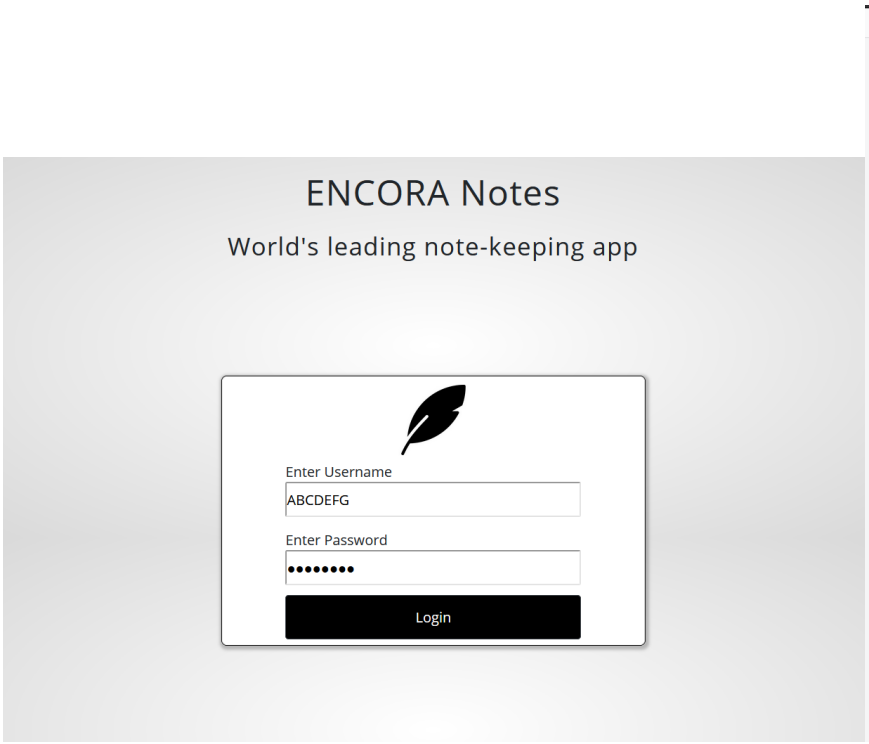
noteid	username	note	title
1	steveb	Once upon a time there was a Hound of Basker...	The Hound of Baskersville
2	steveb	Loren ISpum blah blah balh blah	The Mask
4	steveb	Hola Amigos! 2020 has been a bad year	Hola Amigos
20	steveb	Dummy Note for testing	Dummy Note
21	steveb	Dummy Note for testing	Dummy Note
22	steveb	Note added to the DB	New Note for Demo
NULL	NULL	NULL	NULL

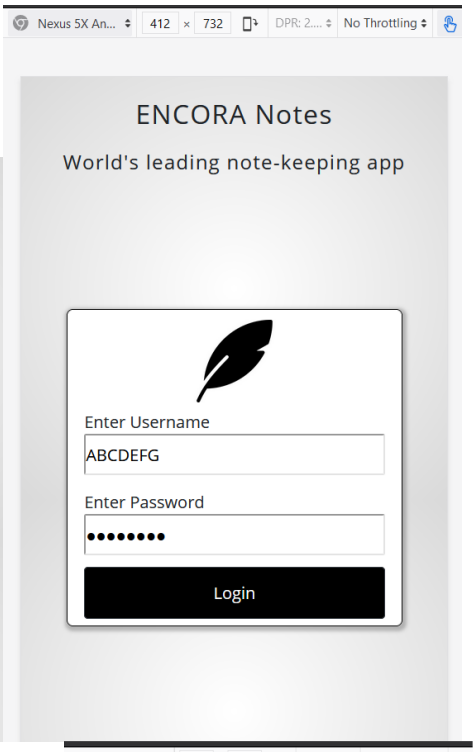
Deletion of notes:
The deletion is performed using the X markers besides the notes:

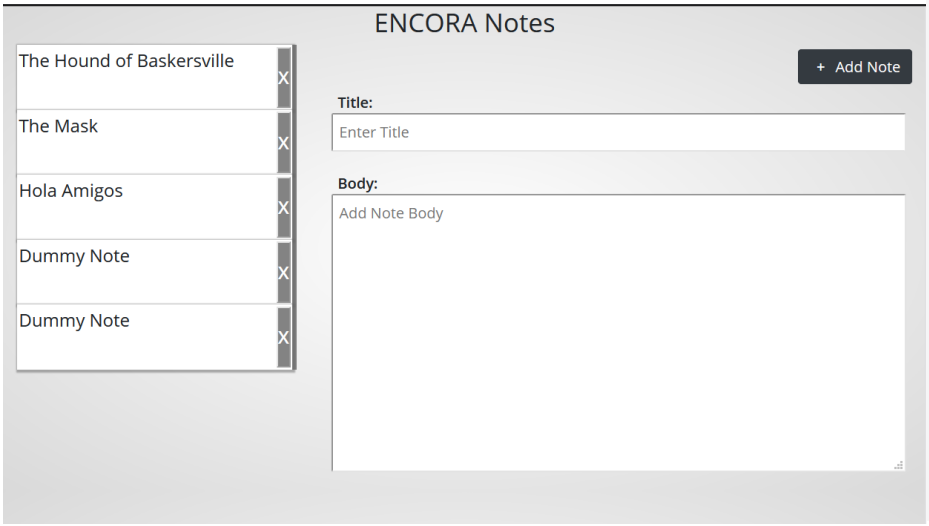


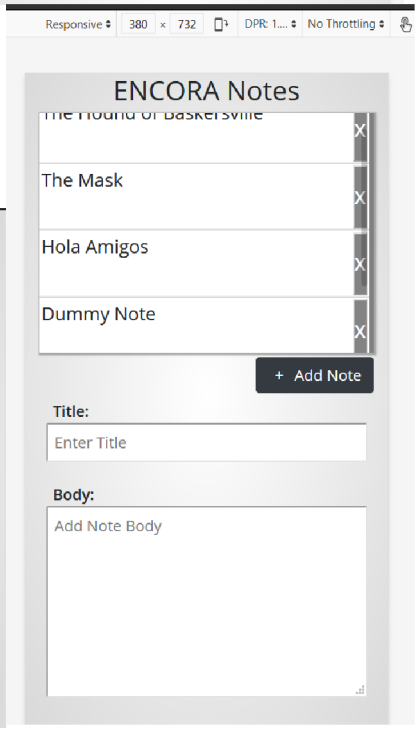
Clicking on these updates the database for that particular user to delete that particular note.

Responsive Mode Screenshots:









The notes-list is completely scrollable (scrollbar is faintly visible in the right screenshot).